Universidad Francisco de Vitoria UFV Madrid

NOMBRE: _____

GRUPO: _____ FECHA: _____

CALIFICACIÓN:

EXAMEN FINAL – PRÁCTICA (60%)

DESARROLLO E INTEGRACIÓN DEL SOFTWARE

Instrucciones

El razonamiento hasta las soluciones tiene la misma importancia que la propia solución. Se valorará positivamente un razonamiento adecuado. Lee detenidamente todo el ejercicio antes de empezar y con ello poder elegir el orden en que debes contestar cada apartado.

La duración de esta parte es de 140 minutos. Y 10 minutos para la entrega.

EJERCICIO P.1 (10 puntos)

Requisitos y pautas generales de trabajo:

1. Acceder a la siguiente URL de Github Classroom: https://classroom.github.com/a/aqABPp3T.
2. Seguir el procedimiento de Github para la creación del repositorio.
3. Clonar el repositorio que ha sido creado para el alumno. El repositorio consta de dos carpetas ***frontend*** y ***backend***. La carpeta frontend contiene dos clases que os ayudarán a mapear el objeto JSON devuelto por SWAPI a objetos Java.
4. Trabajar en el enunciado del ejercicio en el repositorio en local.
    a. **Es obligatorio realizar al menos un commit cada 15 minutos.**
5. Una vez dado por concluido el ejercicio, se deberá:
    a. **Realizar un commit final y crear una release v1.0.**
    b. Comprimir la carpeta de trabajo y subirla a la tarea habilitada en Canvas para tal fin.
    c. **Es imprescindible para dar por válida la prueba incluir la carpeta .git**
    d. Eliminar la carpeta target/ para asegurar que el tamaño del fichero comprimido es razonable y no dé problemas con la subida al aula virtual.
6. Debéis entregar también un fichero de texto plano, llamado **referencias.txt**, en el que tendréis que referenciar todas las fuentes que hayáis usado. Ya sean los repositorios de vuestras prácticas o repositorios propios que tengáis con ejercicios. Asimismo, toda referencia externa, por ejemplo, a StackOverflow, debe ser también referenciada en este fichero. Con poner el enlace a la referencia, es suficiente.

Enunciado

Se pide desarrollar una aplicación web con un frontend en Vaadin y una API RESTful desarrollada con Spring Boot que permita obtener datos de una API externa, manipularlos en el frontend y guardarlos en un fichero JSON mediante una llamada a nuestra API local.

Frontend

El funcionamiento de la aplicación frontend será el siguiente:

En la vista, deberá existir un botón llamado “Obtener naves”, un input llamado “Page”, un Grid y un botón llamado “Guardar”.

El input “Page”, permite indicar el número de página con los resultados que se quieren obtener. Esto es debido al comportamiento de SWAPI, puesto que pagina los resultados devueltos. Si el campo se encuentra vacío o es igual a 1, devolverá los resultados de la primera página. Si, por el contrario, se introduce algún número menor que 5, devolverá los resultados de la página indicada. Si el número no está en ese rango, deberá mostrar una notificación de que la página no es correcta.

Desde el frontend hecho en Vaadin, pulsando en el botón “Obtener naves” y, en base a la información del campo Page, se obtendrán los datos relativos a las naves espaciales contenidas en la API SWAPI y se almacenarán en un grid que se cargará automáticamente con los datos recuperados. Es decir, sin necesidad de recargar la página. Los datos pueden estar repetidos. Si se repite una llamada varias veces, se obtendrán datos repetidos que pueden almacenarse en el ArrayList que alimenta el grid.

La url a la que se deben hacer las peticiones para obtener las naves es la siguiente: *https://swapi.dev/api/starships/?page=%s*, donde %s hará referencia al valor del parámetro “Page” introducido desde la interfaz.

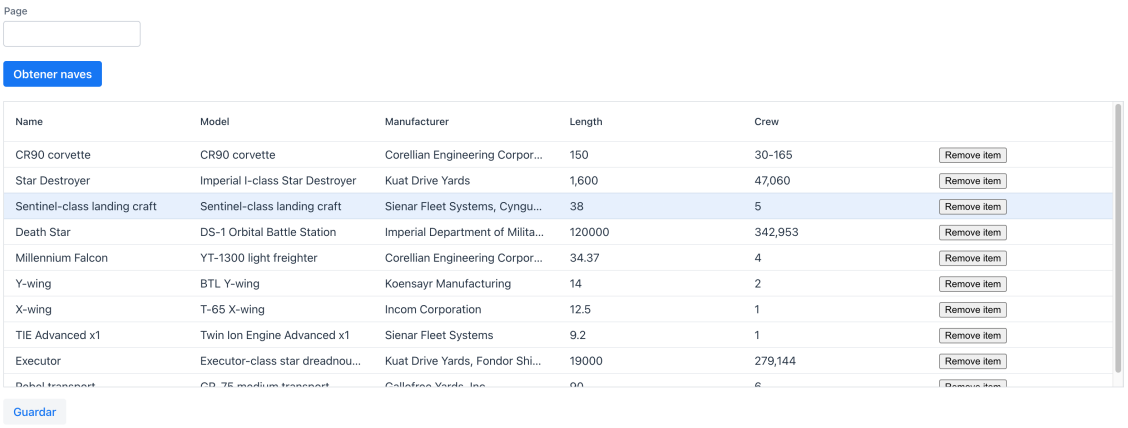
La última columna del grid deberá contener un botón o icono que dé la opción de borrar esa fila y todas aquellas que contengan los mismos datos. Es decir, eliminará todas las apariciones de dicha entrada, sin importar el número de veces que estén repetidas en el ArrayList. [Anexo I].

La manera de comparar si los objetos son iguales es comparar el nombre de estos. Si tienen el mismo nombre, se eliminan. Los datos se actualizarán automáticamente en el grid. **El borrado se hará en el frontend y no en la API.**

Finalmente, se deberá implementar el botón Guardar, que realizará una petición **POST** a la API local implementada por nosotros, enviando el array con todos los elementos disponibles en el grid, para que se escriban en un fichero y se disponga así de una copia local de dichos elementos.

La comunicación con el backend y la API externa SWAPI se realizará mediante llamadas HTTPRequest.

A continuación, disponéis de una pequeña plantilla de la interfaz solicitada:



API REST.

La API RESTful implementada mediante Spring Boot contendrá dos métodos.

Un método POST y un método GET.

- **POST**: Recibirá un array en formato JSON de naves espaciales y los insertará en un fichero JSON. Los datos deben quedar guardados en disco independientemente de la sesión del frontend. Esto es, la primera vez que se llame a este método de la API estará vacío y se insertará la información obtenida. Las siguientes ocasiones en que se llame al método, se deberá añadir siempre la información recibida al contenido existente. O, lo que es lo mismo, **el fichero no debe ser nunca borrado o vaciado una vez se crea** y empieza a recibir contenido.
- **GET**: Devolverá todos los datos almacenados en el JSON y se podrán consultar mediante un cliente tipo ***POSTMAN*** o ***ThunderClient*** de Visual Studio Code.

1) Requisitos y pautas del desarrollo front

1. La versión de Java será Amazon Corretto 11.
2. Utilización del framework **Vaadin** 14.9.1
3. Todas las dependencias deberán ser gestionadas por **Maven**.
4. Para la creación del proyecto, se debe trabajar con la plantilla para Vaadin 14 disponible en el siguiente enlace: https://vaadin.com/hello-world-starters.
5. Dependencias adicionales
    a. GroupID: com.google.code.gson
    b. Artifact ID: gson
    c. Version: 2.10.1
6. **Definir el Group ID del proyecto a "es.ufv.dis.front.final_extraordinaria2022" y el Artifact ID con las iniciales del nombre del alumno.**
7. La interfaz deberá contener el formulario de alta de los datos mencionados anteriormente. Se valorará positivamente un diseño de la UI adecuado.
8. **Comunicación con la API**
    a. Las llamadas a la API se realizarán mediante el uso de HTTP Requests, disponibles en Java 11.
9. Recordad que para recargar un grid sin necesidad de recargar la página, es necesario realizar los siguientes pasos:

a. Actualizar el contenido del ArrayList con los datos obtenidos: aprovechando el método ***addAll()*** que ofrece el ArrayList de Java.
b. Actualizar el contenido del grid mediante la siguiente llamada:

```
grid.getDataProvider().refreshAll();
```

2) Requisitos de la API

Para generar la API, se hará uso de Spring Initializr. Recordad seleccionar la **versión 2.7.13 de Spring** y de marcar **Java 11**, así como **Proyecto Maven**, para garantizar la compatibilidad con vuestra versión de Java.

Definir el Group ID del proyecto a "es.ufv.dis.back.final_extraordinaria2022" y el Artifact ID con las iniciales del nombre del alumno.

Entregables

1. La carpeta de ambos proyectos y la carpeta .git del proyecto.
2. Código de los proyectos alojado en el repositorio de Github.
3. Fichero referencias.txt correctamente cumplimentado.

Se corregirá únicamente por lo entregado en Canvas. No sirve sólo la subida a Github.

Anexo I. Botón "Eliminar"

A continuación, tenéis un pequeño snippet de código para añadir al grid una columna con un botón para eliminar la fila. El método, debéis implementarlo vosotros.

```
grid.addColumn(
        new NativeButtonRenderer<>("Remove item",
                clickedItem -> {
                    // Llamada al método de eliminar
                })
);
```

Calificación del ejercicio

A continuación, se indica cuál es la distribución de las notas de este examen:

Apartado	A	B	GIT
Puntuación	5	3,5	1,5

Universidad Francisco de Vitoria UFV Madrid

NOMBRE: _____

GRUPO: _____ FECHA: _____

Rúbrica de evaluación

	VALORACIÓN			
DIMENSIÓN	**0 puntos**			**10 puntos**
Implementación de los métodos de la API	No implementa los métodos de la API solicitados en el enunciado	Hay una definición de los métodos solicitados, pero no compilan correctamente o presentan errores de funcionamiento.	Los métodos están definidos y realizan la acción solicitada, pero no recuperan los datos correctamente.	Los métodos están correctamente definidos y recuperan la información solicitada correctamente.
Utiliza las funcionalidades de Git de forma correcta	No existe control de versiones	Control de versiones excesivamente sencillo	Utiliza las funcionalidades justas de forma correcta.	Utiliza todas las funcionalidades estudiadas de forma correcta.
Interfaz implementada mediante Vaadin 14.	La interfaz no se ha implementado.	La interfaz existe, pero no dispone de todos los campos solicitados ni es capaz de enviar las peticiones ni mostrar la información.	Existen todos los campos de formulario solicitados, pero existen errores en los tipos o el proceso de envío falla. Los datos se visualizan, pero de forma incorrecta.	La interfaz funciona correctamente, dispone de todos los campos solicitados y permite enviar las solicitudes, así como muestra los datos correctamente.
Uso correcto de la librería HttpRequest de Java 11	No usa la librería HttpRequest de Java 11.	Se hace uso de la librería, pero no corresponde con lo que se solicita en el enunciado. La estructura de los métodos llamados no es correcta.	Se hace uso correcto de la librería, pero existen errores de concepto en la gestión de los métodos o en los datos devueltos.	El uso de los métodos de la librería es el correcto y los datos recuperados son los correctos.
Argumenta correctamente las decisiones de desarrollo y las basa en la experiencia previa.	No basa las decisiones en razonamientos lógicos o relacionados con lo visto en la asignatura.			Las decisiones de diseño e implementación están basadas en razonamientos lógicos desarrollados a partir de los conocimientos adquiridos en la asignatura.

Generación de los datos solicitados, tanto los ficheros como los nuevos objetos	No se generan los ficheros pdf para cada uno de los elementos nuevos.	Los ficheros generados son incorrectos o no coincide el nombre con el especificado. Los objetos no se añaden o se añaden al fichero JSON pero no son correctos.	Los ficheros generados son correctos pero el nombre no coincide. Los objetos se añaden al fichero JSON pero no son correctos.	Todos los ficheros generados son correctos en formato y nombre y los datos se añaden correctamente al fichero JSON.